Project 3: SVM and FDA

By

Valerie A. Rivera

&

Armando Herrera

University of Texas at Rio Grande Valley

CSCI 6366: Data Mining

Fall 2019

November 26, 2019

Abstract

Our assignment involved FDA to find a projection direction so that the two classes of points can be separated. According to the assignments instructions the middle of two projected means that it can be the separating boundary. As indicated in the projects instructions the SVM is also boundary based classifier. Since Support Vectors (SV) are close to the boundary, we can guess the points around the boundary by FDA can contain the SVs.

Therefore this report will consist of Find the points around the boundary determined by FDA for the training of SVM. Report the classification accuracy. Train SVM using different sizes of boundary segments, report accuracies. According to the assignments instructions we are allowed to utilize one pair of classes from the MNIST dataset for training and testing. The first part of the paper will describe Valerie Rivera section of the project. The second part will describe Armando Herreras section of the project.

TABLE OF CONTENTS

SVM Section of the Project

Design:

- Mathematical:

The mathematical design is a Support Vector Machines (SVM). We begin with a Support Vector Machine. It is important to note that because an SVM is a binary classifier, it classifies for, against, or between classes. In our specific design, we created an SVM for every class. My partner and I decided instead of doing many dot operations to do a single matrix operation for every SVM. Therefore, we combined every SVM into a single weight and a bias matrix.

- Linear SVM classifier $\vec{w} \cdot \vec{x} + b$
- A multi-class SVM classifier $\bar{w} \ \cdot \vec{x} + \vec{b}$

A fourier approximation of the Radial Basis function kernel was used to increase the dimensionality of the hyper-plane that was fitted to the data.

$$K(x, x') = e^{-\gamma \|x - x'\|^2}$$

The fourier approximation can be found in the paper, “Random Features for Large-Scale Kernel Machines” by Ali Rahimi, et al, (2007). In the paper, in Algorithm 1, the author describes the algorithm by which the Radial Basis Kernel function can be approximated.

Implementation:

Two implementations introduced into our project is the loss function, which is cross-entropy. The optimization algorithm used to minimize our weight and biases is known as stochastic gradient descent.

We created a weight PyTorch matrix and a biases PyTorch vector, which holds a distinctive characteristic which records changes for gradient calculation. This characteristic comes into play when one performs operations on a PyTorch object. The PyTorch object remembers the processes, which are later used to backpropagate through our matrix multiplication, our addition operation, and our loss function. This helps in turn to minimizes the error of the SVM. We are, therefore training it to classify MNIST. The kernel approximation increased the dimensionality from 784, which is the number of pixels in an MNIST image, to 940, which was found empirically. The γ hyper-parameter was also found empirically and was set to 0.004. The approximation required the generation of random weights in a Gaussian Distribution, with a standard deviation of 1 and a mean of 0, multiplied by $\sqrt{2\gamma}$. An offset was also precalculated with some random numbers except with a uniform distribution between 0 and 1 then multiplied by 2π . The weights and the bias matrix and vector were also calculated in a similar fashion as to avoid having to use the kernel approximation on it as well. So,

$$\bar{w}\phi(x) + \vec{b}$$

Where ϕ is our kernel approximation and is defined by

$$\sigma(x) = \tfrac{2}{N} \cos(\vec{x}\vec{k}_i + o_i)$$

Where N is the number of output dimensions, k and o are the randomly generated set of number.

Settings:

The optimal settings that we found were with the learning rate at 0.005 and training the SVM for six epochs. This means that we trained the network over all of the data sixteen times. When the epoch would finish, the input data were randomly shuffled. All the input images were scaled from between 0 to 255 to -1 to 1, which is a range of 8-bit integers, and converted to single-precision floating-point numbers.

Results:

We achieved an accuracy of 92.48 percent in our MNIST testing dataset. We saved the SVM to a file for future use.

FGSM Section of the Project

Design:

An adversarial attack used to generate samples is called *Fast Gradient Sign Attack (FGSM)* and is described in a paper titled, “Explaining and Harnessing Adversarial Examples”. The paper shows that by calculating the gradient, $\nabla_x J(\theta, x, y)$, through a model, neural network or SVM, of an image, through backpropagation, and by adding the sign of the gradient multiplied by a hyper-parameter, ε, we can decrease the accuracy of models that relies heavily on linear elements.

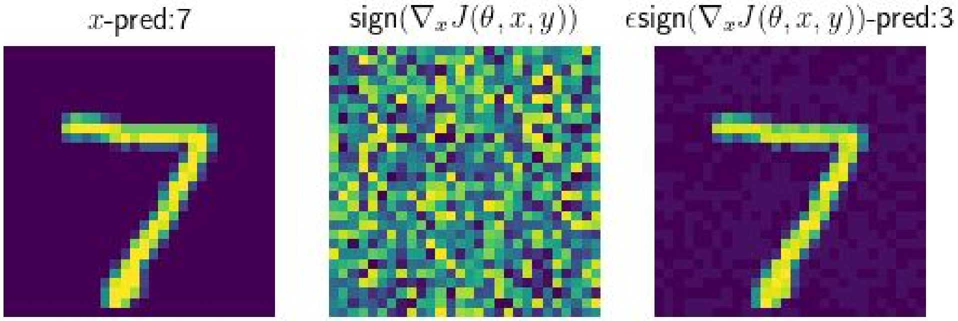
$$f(x) = x + \varepsilon sign(\nabla_x J(\theta, x, y))$$

Settings:

The goal for the epsilon hyper-parameter was to set low enough for it to produce samples with barely recognizable changes and still fool the SVM. An epsilon value of 0.08 was chosen.

Results:

The result was a PyTorch system that brings down the classification accuracy to 10.92%.



x-pred:7 sign$(\nabla_x J(\theta, x, y))$ ϵsign$(\nabla_x J(\theta, x, y))$-pred:3

The leftmost image is of a correctly classified image of a seven classified as a seven. The middle image is of the gradient map created from back-propagating through the SVMs. The rightmost image is of the perturbed image which was incorrectly classified as a three instead of what it is, a seven.

SVM w/ FDA Section of the Project

Design:

Our use of the Fisher Linear Discriminant Analysis (FDA) for multiple class is slightly different from that of the two classes. To calculate the Scatter matrix we use the following function:

$$S_i = \sum_{x_j \in Class\ i} (x_{ij} - \mu_i)(x_{ij} - \mu_i)$$

Where x_{ij} is the sample from i th class and j he sample in that class and μ_i is the mean of all the samples in i th class. The within scatter matrix for all classes is then calculated by summing the scatter matrices for all classes in the mnist dataset.

$$S_W = \sum_{i=1}^{c} S_i$$

Where c is the number of classes. Then, using the inverse of S_W we calculate the discriminant for each class against all of the rest.

$$v_i = S_W{}^{-1}(\mu_i - \tfrac{1}{c-1}(\sum_{j=0}^{i-1} \mu_j + \sum_{j=i+1}^{c} \mu_j))$$

Then to calculate the distance from the line on a per class basis.

$$y_i = v^t c_i{}^t$$

Implementation:

To implement FDA only the numpy linear algebra library was used. All samples were separated into their respecting class, with a list of numpy n-dimensional array, one for each class. We also counted the number of samples per class here. Then, in the same loop, the mean for each class was calculated and that class a scattered matrix and summed. The pseudo-inverse of the summed scatter matrix is then calculated using a numpy function. In another loop the discriminant line for each class is calculated. Then all samples are passed to find their distance from the discriminant lines and the output is normalized so that their distances are between 0 and 1. A filter is then done to filter all samples more than a predefined hyper-parameter between 0 and 1. This way only the points closer the discriminant lines are kept.

Settings:

An additional cut-off floating point number is used to as a hyper-parameter for the cutt-off of the FDA. We found that the optimal value is 0.2284 resulting in about half of the samples pruned. With the additional FDA the other most optimal values are a learning rate of 0.09 with a gamma of 0.003339907, the number of kernel dimensions used is 834, and the SVM was trained for 14 epochs.

Results:

We achieve an accuracy of 92.98 percent with FDA. The time that it took the modified code was 213 seconds.

CODE

https://github.com/anhydrous99/SVM

License Link:

https://github.com/anhydrous99/SVM/blob/master/LICENSE

Conclusion

We learned to build an SVM that classifies images of handwritten numbers.
The limitations of this project were our time limitation and the limitation of full knowledge of SVMs. A considerable result was substantially decreasing training time of the SVM with FDA. A foreseeable avenue is to pursue whether an SVM trained with FDA filtered data is as generalized as an SVM trained on all data.